SMARTBRIDGE
Let's Bridge the Gap

Intership Report
On
Smart Agriculture System
Based On IoT

Table of Contents

1. Introduction

1.1 Overview

Agriculture plays a important role in country's economy and provides a large-scale employment to the people. However, agriculture is highly dependent upon weather and climate. For example, changes in temperature, soil moisture, and humidity may result in low yield of crops. It is significant to monitor environmental parameters in order to manage crop growth and increase the agricultural production yield. The sensed information very important for decision making.

In the modern world, every appliances humans use are connected to the internet. Controlling machines and other useful devices via the internet facilitates less time consumption and ease in control over the appliances. This project focuses on automation of the agricultural works carried out in the field by live monitoring and access to control by various IoT devices via the internet. The eye-catching features of this project include smart irrigation with smart control based on real time field data.

1.2 Purpose

- Smart Agriculture System based on IoT can monitor soil moisture and climatic conditions to grow and yield a good crop.
- The farmer can also get the realtime weather forecasting data by using external platforms like Open Weather API.
- The mobile app can monitor the temperature, humidity and soil moisture along with weather forecasts.
- Based on all parameters one can water his crop by controlling the motors using the mobile application from anywhere.

2. Literature Survey

2.1 Existing Problem

Human beings witness extreme climatic changes, deteriorating soil fertility, dry lands and collapsed ecosystem these days. The increasing population and decreasing cultivable lands and farmers make will surely make people's future worse in case of healthy and sufficient food. To encourage agriculture even when engaged in a full profession other than field work, and to facilitate the farmers an easy manipulation of the manual works that need to be done crucially. Moreover, not being able to adapt to technological trends has become fatal to agriculture.

The challenges of a smart agriculture system include the integration of these sensors and tying the sensor data to the analytics driving automation and response activities. Ideally, each field should get just the right amount of water at just the right time. Under-watering causes crop stress and yield reduction. Over watering can also cause yield reduction and consumes more water and fuel than necessary and leads to soil erosion and fertilizer, herbicide, and pesticide runoff.

2.2 Proposed Solution

There are many new farming techniques like precision farming, GSM based farming and smart irrigation control. Here, we propose a solution using cloud and IoT to monitor the soil and weather conditions. The temperature, humidity and soil moisture sensors are used to obtain necessary information and push them to the cloud platform. Furthermore, we create a web page which is accessed by the farmers to monitor their crops and control motors to water the crops.

The android based Farming system is an automatic irrigation system which performs multiple operations in agriculture; this project uses a centralized controller which is programmed to receive the input signal of multiple sensors of the field. Based on these real time field parameters, motor pump can be controlled to water the fields.

For large-area, more traditional farming, sensors placed within the ground may record real-time data on soil moisture, temperature and pH, while environmental sensors may record sun exposure, rainfall, wind speed, air temperature and humidity. Aerial drones may also be used for surveillance of crops and pests.

For smaller, indoor farming, LED lighting, precise control of photo-period, and soil and environmental sensors can reduce the cost of energy and increase yields.

3. Theoretical Analysis

3.1 Block Diagram

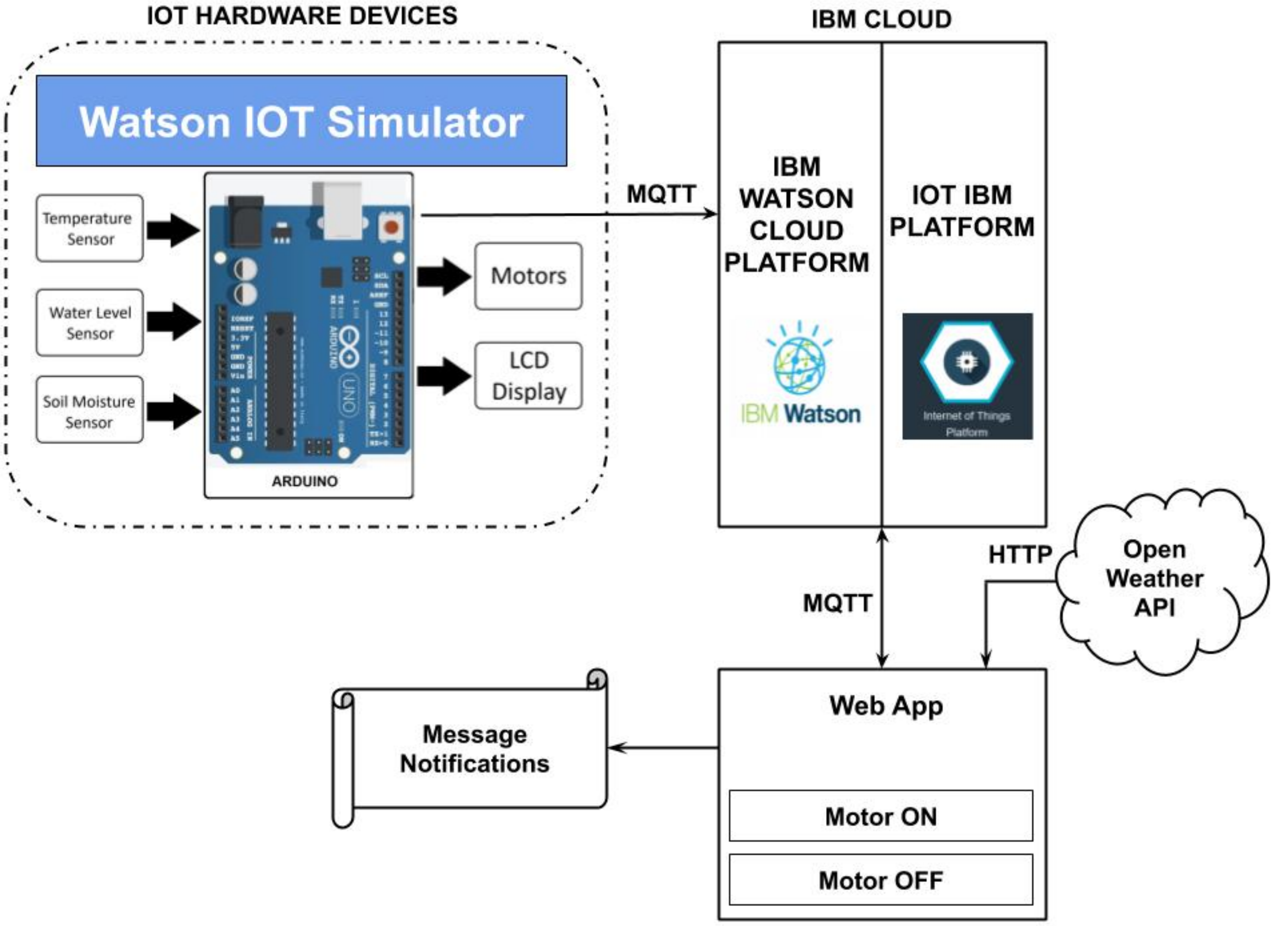

Fig 3.1 Block Diagram

The basic idea behind the project is that the sensors in the field will relay the data to IBM Cloud which in turn will send the data to the Web Application. For the project instead of using actual sensors and hardware we are using Watson IoT Simulator to provide use data and simulate the whole setup. The Web Application displays data such as the Temperature, Humidity and Soil Moisture. Once the soil moisture starts getting low, the system sends out a message to the user about low moisture level.

The user can also see live weather details of the location which is being continuously updated by Open Weather API. Depending on the weather and soil conditions, the user can switch on the motors remotely from anywhere. Once the motor is switched on a timer starts and sends out a warning message after 30 minutes of usage and automatically switches off the motor after 45 minutes of usage and sends a message to the user about the same.

3.2 Hardware/Software Designing

Hardware Designing

The Arduino device is connected with the following three sensors:-

- ✓ Temperature Sensor - LM35
- ✓ Humidity Sensor - DHT22
- ✓ Soil Moisture Sensor - VH400

The Arduino device is connected to the IBM Platform which receives the data from the sensors. For the project we are using Watson IoT Simulator to simulate the Arduino device and the sensors connected to it. The motors are also connected to the IBM Platform to receive the commands ON/OFF.

Software Designing

For the project node.js software is used. NodeRed is an application of node.js used in software designing part. For receiving input data we use the Watson IoT Simulator. Two devices namely, 'NodeMCU' and 'Motor' are created in the IBM Platform. 'NodeMCU' device is linked to the Watson IoT Simulator to receive the device events and 'Motor' device is linked to the motors and receive device commands from the NodeRed UI. NodRed is a programming tool for wiring hardware devices, online APIs. It facilitates browser based editor which makes connecting the flows much easier.

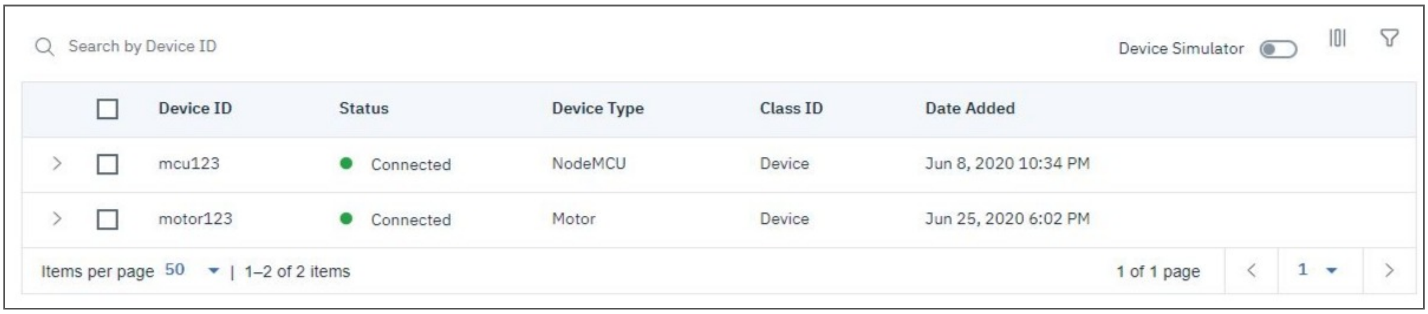

Fig 3.2 Devices made in IBM Platform

Nodes of node-red such as IBM IOT IN/OUT and Open weather plays crucial roles in this project as they are directly concerned with the input and output of the entire project. JSON language is used to send messages as payload to the devices in and out in this project.

IBM Watson IOT Platform is a foundational cloud offering that can connect and control IOT sensors, appliances, homes, and industries. Built on IBM Cloud, Watson IOT Platform provides an extensive set of built-in and add-on tools. Use these tools to process IOT data with real-time and historical analytics, extract key performance indicators (KPIs) from your data, add “smarts” in the cloud for non-smart products, and securely connect your own apps and existing tools to the Watson IOT Platform infrastructure.

These two devices are connected to IBM in and IBM out node of the nodeRed editor respectively. Using generated api from the platform this measure to connect the devices were done. The IBM in node was given event type and the IBM out node was given Command type which makes them ideal in the full pledged working of this application.

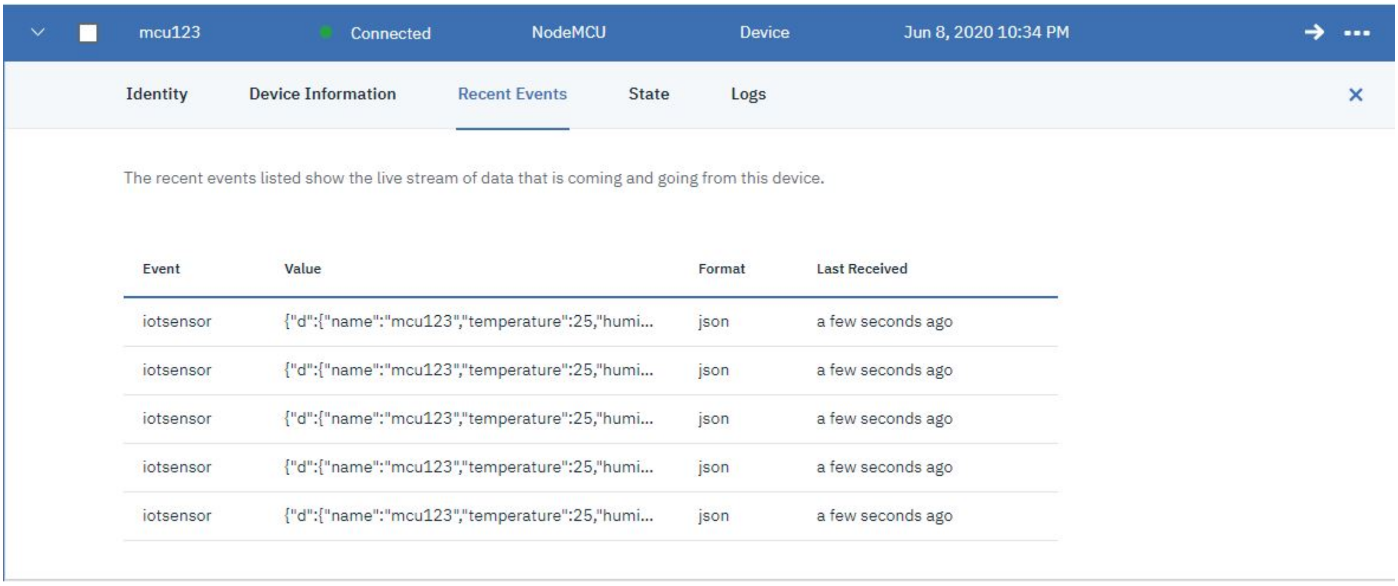

Fig 3.3 Device events received by NodeMCU

The NodeRed also uses a timer which starts as soon as the 'Motor ON' button is pressed and sends out a warning message after 30 minutes and sends out a warning message at the end of 45 minutes of timers and also switches off the motor by sending the command to the 'Motor' device.

The messaging service is being provided out by Twilio API. A message is also sent by the program as soon as the soil moisture level starts going below the ideal conditions for the crop.

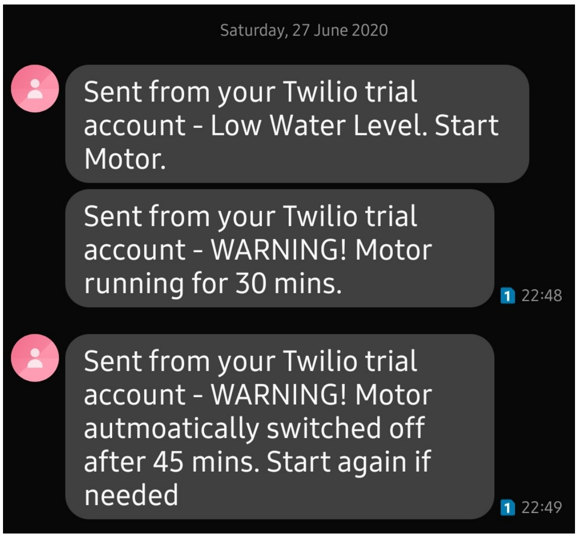

Fig 3.4 Snapshot of notification received from Twilio API

4. Experimental Investigations

The experiments were carried out in node-red dashboard using the IBM Watson simulator which provided the parameters temperature, humidity and object temperature (soil moisture). The motor controls are set to turn on and off by the use of buttons. When triggered a command, the python code which was set to catch information from the device's output responded correctly. This shows that the command from the online dashboard controls the motors and lights on time.

The connection to the IOT simulator from the blue mix requires a Organization ID, which can be found at the top right corner of the Watson IOT platform. Furthermore, the ID of the device which is going to receive the data from simulator and access token are necessary to connect to the simulator.

As soon as the device is connected the name appears at the top right corner of the online simulator. Also in the IOT platform, the device status is connected. After a few seconds' data as payload messages in JSON format will be sent to the device. These messages can be viewed at the recent events column.

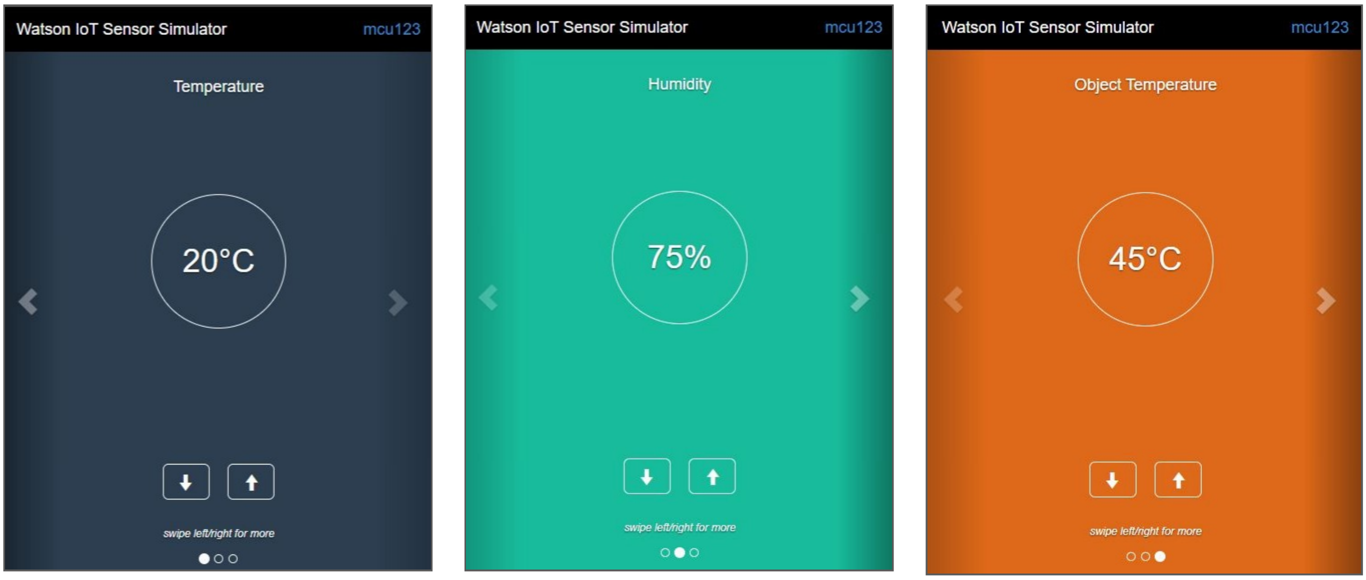

Fig 4.1 Watson IoT Sensor Simulator

The application is made in such a way that the user can set the optimal conditions for the crop in the application. As soon as the Soil moisture parameter starts going low and crosses the optimal threshold, the system sends out an SMS notification to the user warning about the low water level and the need to start the motor. SMS notification features are added by the help of Twilio API service.

The average optimal ranges for all the three parameters are as follows:-

- Temperature - 20 to 35 °C
- Humidity - 40 to 70%
- Soil Moisture - 40 to 70%

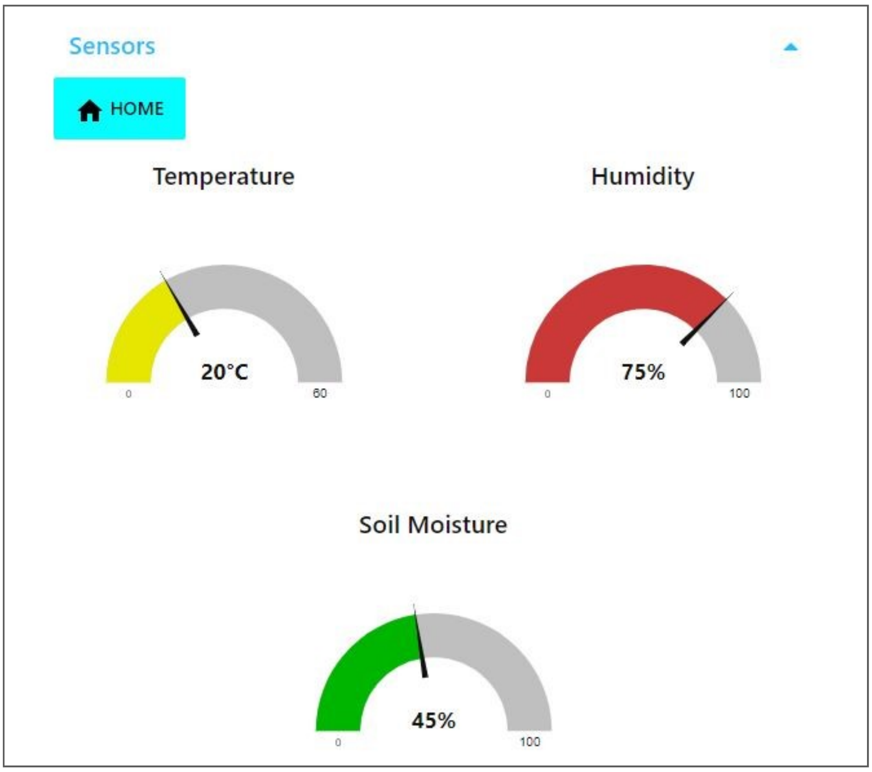

Fig 4.2 Sensor UI

The controls are given as buttons which send command messages to the device central unit and access the irrigation motors, light facilities. The NodeRed also uses a timer which starts as soon as the 'Motor ON' button is pressed and sends out a warning message after 30 minutes and sends out a warning message at the end of 45 minutes of timers and also switches off the motor by sending the command to the 'Motor' device.

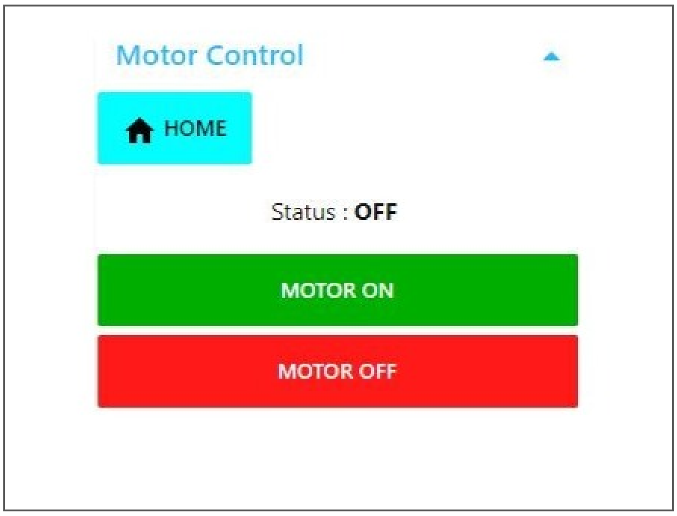

Fig 4.3 Motor control UI

The data from the IBM IoT OUT node from NodeRed which is connected to the second device Device2, receives the commands from the User Interface Dashboard such as light on, off Motor on, off. At the python editor, coding was made in such a way it receives data from the second device by giving the device credentials. When run it showed device connected. And further changes were noted as the commands like motor on, off were given.

The python terminal received desirable output “MOTOR ON” and “MOTOR OFF” as a response to Motor ON and Motor OFF buttons in the UI.

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\sunny\Desktop\Sunny\SmartInternz IBM\IBMsubscribe.py ====
2020-06-28 22:02:17,108   ibmiotf.device.Client      INFO    Connected successfully: d:gmxmw0:Motor:motor123
Command received: {'command': 'motoron'}
MOTOR ON
Command received: {'command': 'motoroff'}
MOTOR OFF
```

Fig 4.4 Python Terminal

5. Flowchart

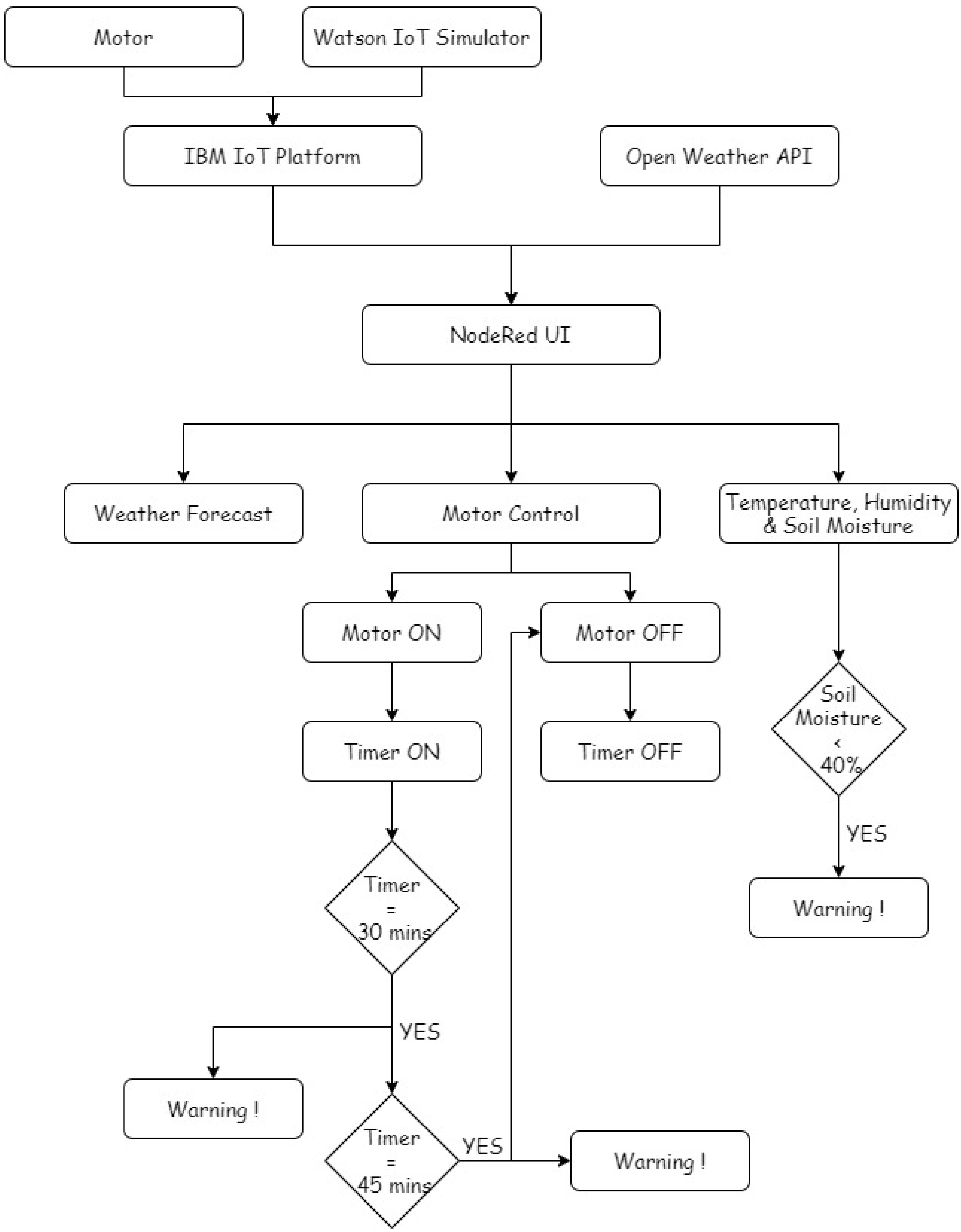

Fig 5.1 Flowchart

6. Result

The changes in the simulator are clearly documented and displayed using the gauges in the dashboard UI. Using Open weather API, the weather forecast of the location was successfully retrieved and displayed in the application. The application gives access to command motor devices present in the field, making irrigation easy and more accurate. All the changes taking place in the application are happening in real time.

Th application is made even more smart by employing SMS services by Twilio API. The application sends out SMS to the user whenever the soil moisture starts getting low. One more advantage is the use of timer in the application making the whole setup electricity efficient. The sensors deployed in the field helps in better irrigation and the timer allows the motors to run for maximum 45 minutes at a stretch preventing over-watering and water wastage.

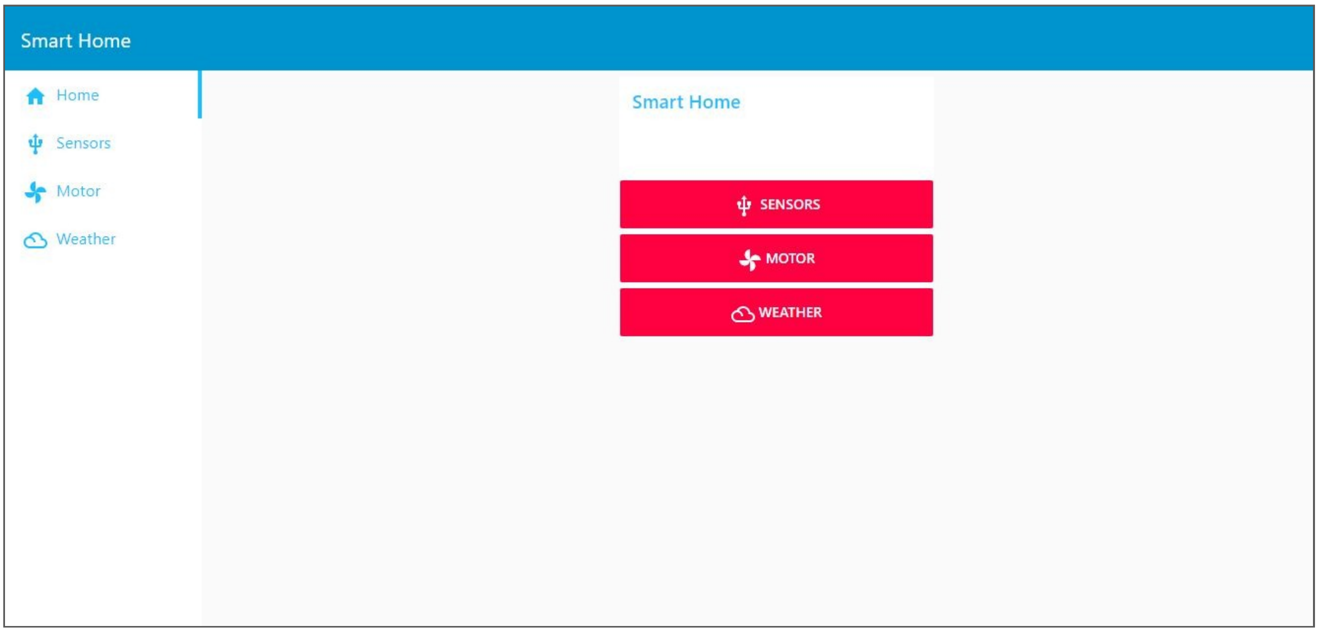

Fig 6.1 Snap of Smart Home page

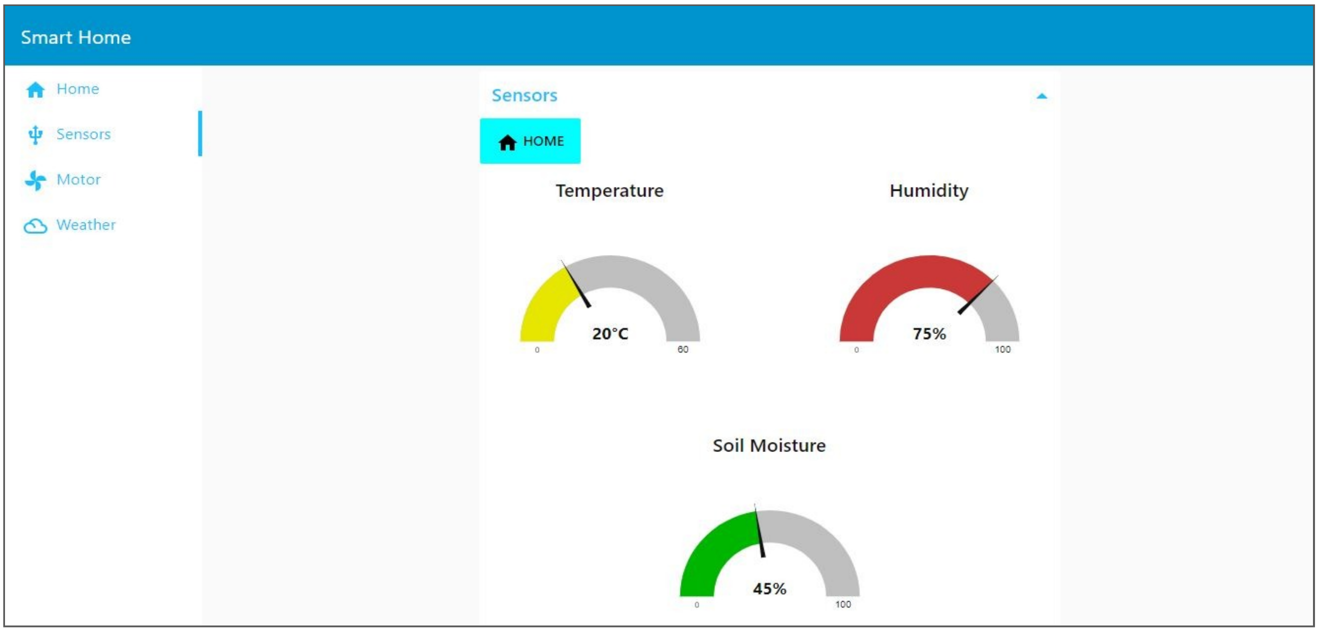

Fig 6.2 Snap of Sensors page

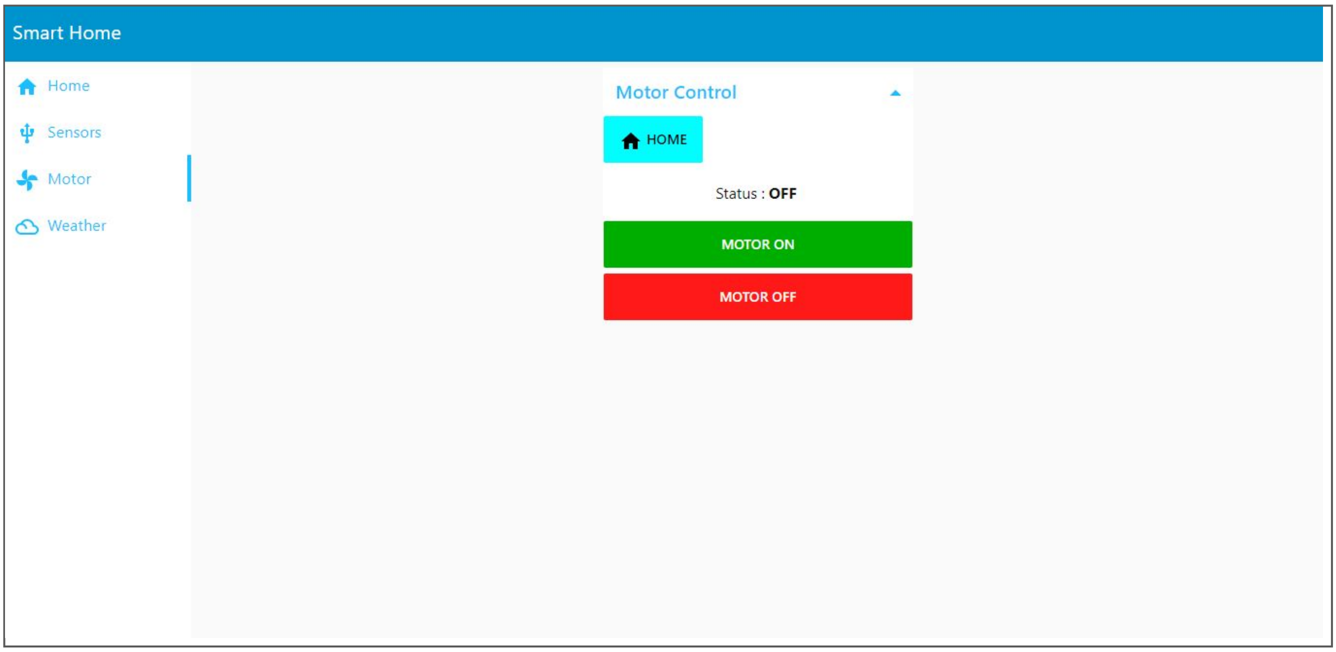

Fig 6.3 Snap of Motor page

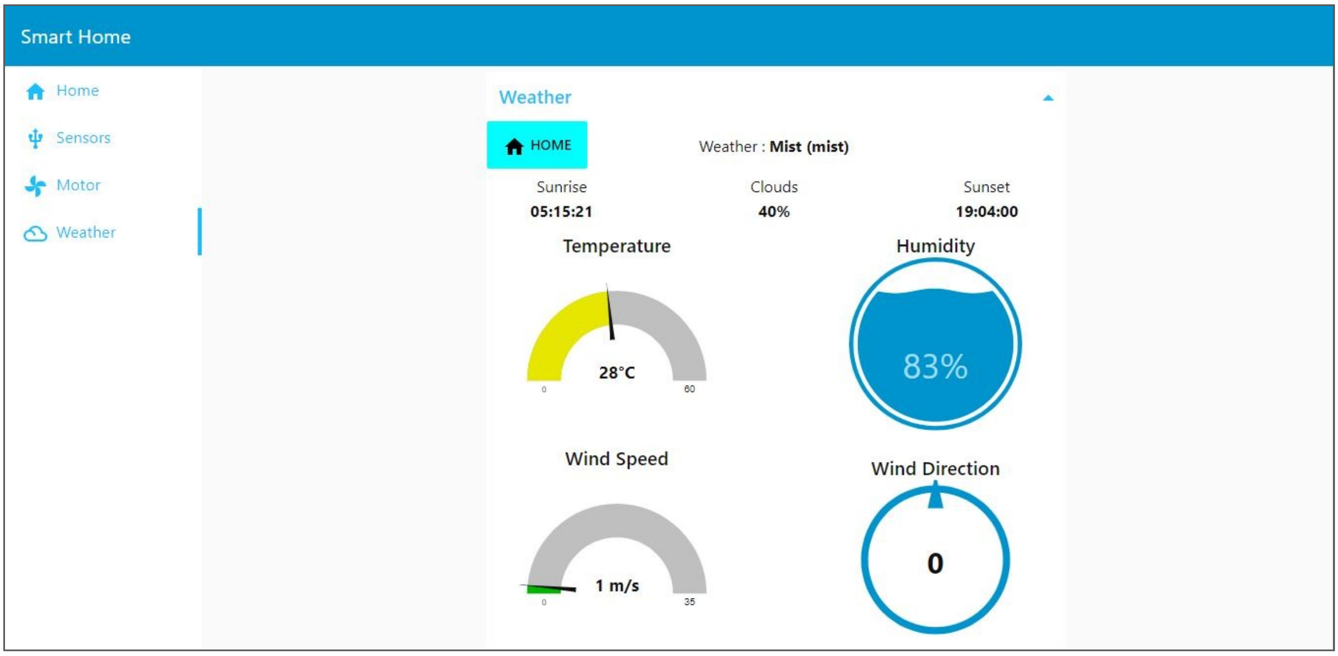

Fig 6.4 Snap of Weather page

7. Advantages & Disadvantages

Advantages

1. It allows farmers to maximize yields using minimum resources such as water, fertilizers, seeds etc.
2. The system increases the crop productivity and reduces farmer's workload.
3. Mobile operated motors save cost of electricity.
4. This automation system will be handy, especially for farmers with farms and fields at a long distance.
5. Farmers can set the time, when the motor turns off automatically.
6. Low soil moisture notifications are beneficial in case farmers forget to water their fields.
7. Live weather forecasts allow farmers to act according to the weather
8. Efficient water usage
9. It is cost effective method.

Disadvantages

1. The smart agriculture needs availability of internet continuously.
2. The smart farming based equipment requires farmers to understand and learn the use of technology. This is major challenge in adopting smart agriculture farming at large scale across the countries.
3. A Farmer has to keep a frequent watch over the application to check for water levels and motors.

8. Applications

- It is very useful in greenhouse farming where every parameter is very intelligently monitored.
- Precision farming makes the farming practice more precise and controlled by realizing smart farming applications such as livestock monitoring, vehicle tracking, field observation, and inventory monitoring.
- The Ground and Aerial drones can be used for assessment of crop health, crop monitoring, planting, crop spraying, and field analysis.
- Real time weather conditions can be used to choose the right crops which can grow and sustain in the particular climatic conditions.
- Large farm owners can utilize wireless IoT applications to collect data regarding the location, well-being, and the health of their cattle.

Fig 8.1 Applications of Smart Agriculture

9. Conclusion

The main concern of this project was to facilitate a farmer with modern methods of controlling the farm land with ease by the use of mobile applications. The various parameters like temperature, humidity etc were monitored using web applications. The data from the weather station like wind speed, temperature, humidity etc were displayed in the web browser. The motor device can also controlled by the web application. The purpose was met satisfyingly and, in future by further development of technologies a much easier task handling mobile application, can be created.

The high efficiency of integrated agriculture production systems delivers socio-economic and ecological benefits that benefit farmers as well as the whole society. There are many ways in which integrated agriculture production systems can help producers to adapt to climate change and provide important mitigation co-benefits. However, several factors hamper the effective adaptation of integrated production systems, such as lack of data on the impacts of climate change, and high requirements in terms of knowledge and labor and initial investments that may pay off only over long time periods.

10. Future Scope

We can further develop the interface screen in order to display the status of the soil nutrient levels, percentage of water utilized to water the plant, duration of time for which the water pump is ON, etc. We can also show the graphical representation of the moisture content levels in the soil. A long range weather forecast can also be added to provide accurate information about the weather. IOT sensors are capable of providing farmers with information about crop yields, rainfall, pest infestation, and soil nutrition are invaluable to production and offer precise data which can be used to improve farming techniques over time. With a future of efficient, data-driven, highly-precise farming methods, it is definitely safe to call this type of farming smart. We can expect IOT will forever change the way we grow food.

11. Bibliography

I. S. S. Gill, I. Chana, C. Science, and R. Buyya, "IoT Based Agriculture as a Cloud and Big Data Service : The Beginning of Digital India," vol. 29, no. 4, pp. 1–23, 2017.

II. R. K. Kodali and A. Sahu, "An IoT based Weather Information Prototype Using WeMos," no. 1, pp. 612–616, 2016.

III. I. Mat, M. Rawidean, M. Kassim, A. N. Harun, I. M. Yusoff, and M. Berhad, "Smart Agriculture Using Internet of Things," 2018 IEEE Conf. Open Syst., pp. 54–59, 2018.

IV. S. R. Prathibha, A. Hongal, and M. P. Jyothi, "IOT BASED MONITORING SYSTEM IN SMART AGRICULTURE," pp. 5–8, 2017.

V. Y. Bo and H. Wang, "The Application of Cloud Computing and The Internet of Things in Agriculture and Forestry," pp. 168–172, 2011.

VI. M. K. Gayatri, "Providing Smart Agricultural Solutions to Farmers for better yielding using IoT," no. Tiar, pp. 40–43, 2015.

VII. Smart Agriculture System using IoT Technology- Muthunoori Naresh, P Munaswamy.

12. Appendix

Appendix A - Source Code

```
import time
import sys
import ibmiotf.application #To install pip install ibmiotf
import ibmiotf.device

#Device details
organization = "gmxmw0"
deviceType = "Motor"
deviceId = "motor123"
authMethod = "token"
authToken = "motor123"

#Callback function
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data)
    if cmd.data['command']=='motoron':
        print("MOTOR ON")

    elif cmd.data['command']=='motoroff':
        print("MOTOR OFF")

    if cmd.command == "setInterval":
        if 'interval' not in cmd.data:
            print("Error - command is missing required information:
'interval'")
        else:
            interval = cmd.data['interval']

    elif cmd.command == "print":
        if 'message' not in cmd.data:
            print("Error - command is missing required information:
'message'")
        else:
            output=cmd.data['message']
            print(output)

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
"auth-method": authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()
```

```
#Connect the device and application from cloud
deviceCli.connect()

while True:

    deviceCli.commandCallback = myCommandCallback

#Disconnect the device and application from cloud
deviceCli.disconnect()
```

https://github.com/SmartPracticeschool/llSPS-INT-2791-Smart-Agriculture-system-based-on-IoT/blob/master/IBMsubscribe.py

Appendix B - NodeRed Flow

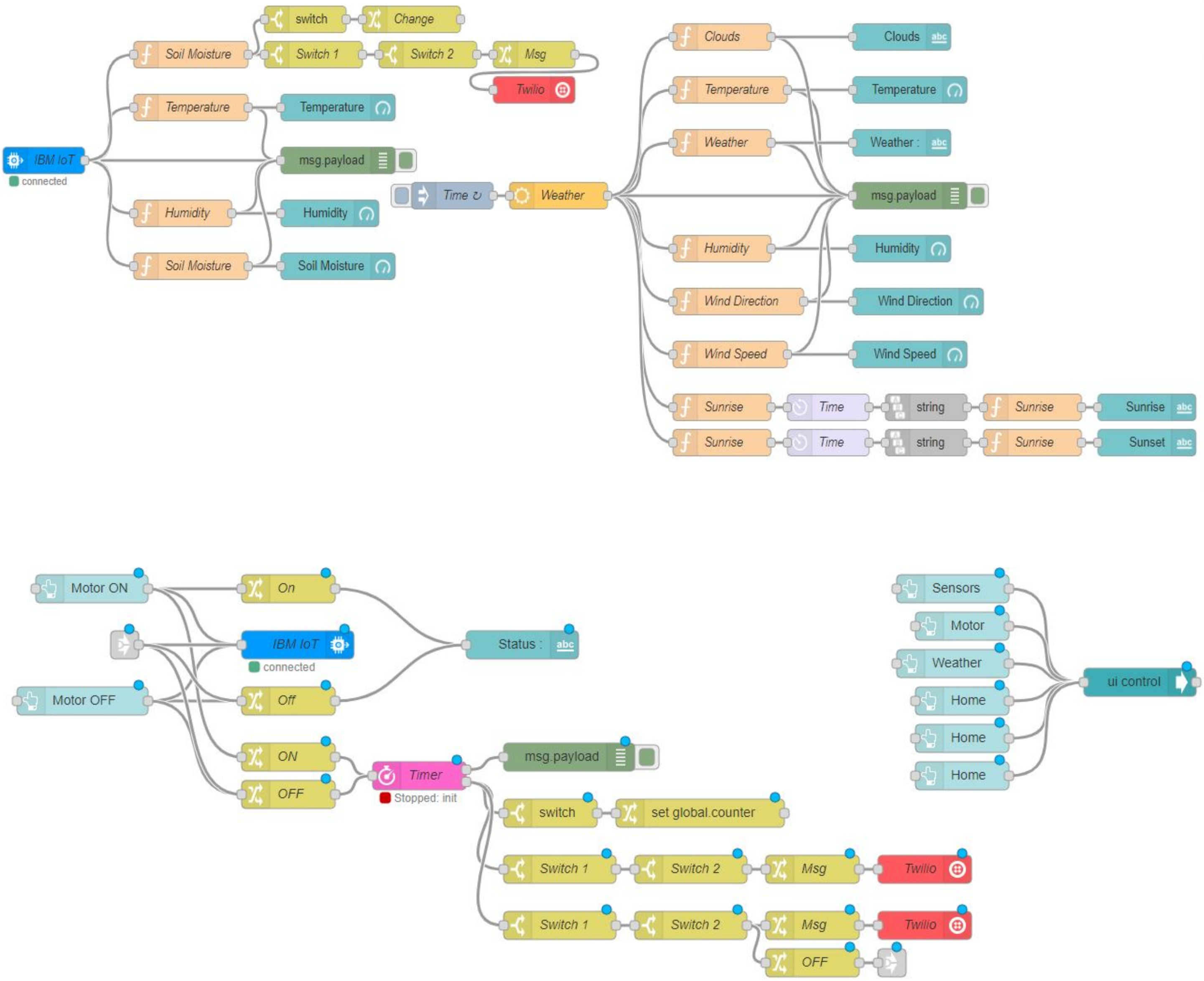

Fig 12.1 NodeRed Flow

https://github.com/SmartPracticeschool/llSPS-INT-2791-Smart-Agriculture-system-based-on-IoT/blob/master/Smart%20Agriculture%20App.json